To appear in the 2005 International Symposium on Microarchitecture (MICRO-38)

A Quantum Logic Array Microarchitecture: Scalable Quantum Data Movement and Computation

Tzvetan S. Metodi[†], Darshan D. Thaker[†], Andrew W. Cross[‡]
Frederic T. Chong[§] and Isaac L. Chuang[‡]
[†]University Of California at Davis, {*tsmetodiev, ddthaker*}*@ucdavis.edu*
[§]University Of California at Santa Barbara, *chong@cs.ucsb.edu*
[‡]Massachusetts Institute of Technology, {*awcross, ichuang*}*@mit.edu*

Abstract

Recent experimental advances have demonstrated technologies capable of supporting scalable quantum computation. A critical next step is how to put those technologies together into a scalable, fault-tolerant system that is also feasible. We propose a Quantum Logic Array (QLA) microarchitecture that forms the foundation of such a system. The QLA focuses on the communication resources necessary to efficiently support fault-tolerant computations. We leverage the extensive groundwork in quantum error correction theory and provide analysis that shows that our system is both asymptotically and empirically fault tolerant. Specifically, we use the QLA to implement a hierarchical, array-based design and a logarithmic expense quantum-teleportation communication protocol. Our goal is to overcome the primary scalability challenges of reliability, communication, and quantum resource distribution that plague current proposals for large-scale quantum computing. Our work complements recent work by Balenseifer et al [1], which studies the software tool chain necessary to simplify development of quantum applications; here we focus on modeling a full-scale optimized microarchitecture for scalable computing.

1. Introduction

Quantum computation exploits the ability for a single quantum bit, a qubit, which can be implemented by the polarization states of a photon or the spin of a single atom, to exist in a superposition of the binary "0" and "1" states (simply denoted as $\alpha|0\rangle+\beta|1\rangle$, where α and β are probability amplitudes satisfying $|\alpha|^2+|\beta|^2=1$). With N qubits a quantum computer can be in 2^N unique states at any given time. These states can be inter-correlated such that a single logic gate can act on all possible 2^N states. The exponential speedup offered by quantum computing, based on the ability to process quantum information through gate manipulation [2], has led to several quantum algorithms with substantial advantages over known algorithms with traditional computation. The most significant is Shor's algorithm for factoring the product of two large primes in polynomial time. Additional algorithms include Grover's fast database search [3]; adiabatic solution of optimization problems [4]; precise clock synchronization [5]; quantum key distribution [6]; and recently, Gauss sums [7] and Pell's equation [8].

A relevant large-scale quantum system must be capable of reaching a system size of $S=KQ\geq 10^{12}$, where K denotes the number of computational steps and Q denotes the number of computational units. Quantum data is inherently very unstable, which leads to a lack of reliable operations that can be performed on it. Also if left idle, this quantum data will interact with its environment and lose state, a process called *decoherence*. Finally, there is the difficulty of transmitting quantum data between computational units without losing state. This implies that the greatest challenge towards a large, practically useful quantum computer, is designing an architecture that incorporates the required amount of fault-tolerance while minimizing overhead.

Previous work in large-scale quantum architecture [9, 10, 11] has led to the consideration of several main scalability issues that must be taken into account: reliable and realistic implementation technology; robust error correction and fault-tolerant structures; efficient quantum resource distribution.

1. Reliable and realistic implementation technology: There are multiple approaches from very diverse fields of science for the realization of a full-scale quantum information processor. Solid state technologies, trapped ions, and superconducting quantum computation are just a small number of many physical implementations currently being studied. No matter the choice, any technology used to implement a quantum information processor must adhere to four main requirements [12]: **1)** It must allow the initialization of an arbitrary *n*-qubit quantum system to a known state. **2)** A universal set of quantum operations must be available to manipulate the initialized system and bring it to a desired correlated state. **3)** The technology must have the ability to reliably measure the quantum system. **4)** It must allow much longer qubit lifetimes than the time of a quantum logic gate. The second requirement encompasses multi-qubit operations; thus, it implies that a quantum architecture must also allow for sufficient and reliable communication between physical qubits.

2. Robust error correction and fault tolerant structures: Due to the high volatility of quantum data, actively stabilizing the system's state through error correction will be one of the most vital operations through the course of a quantum algorithm. Fault tolerance and quantum error correction constitute a significant field of research [13, 14, 15, 16, 17, 18] that has produced some very powerful quantum error correcting codes analogous, but fundamentally different from their classical counterparts. The most important result, for our purposes, is the Threshold Theorem [17], which says that an *arbitrarily reliable* quantum gate can be implemented using only *imperfect gates*, provided the imperfect gates have failure probability below a certain *threshold*. This remarkable result is achieved through four steps: using quantum error-correction codes; performing all computations on encoded data; using fault tolerant procedures; and recursively encoding until the desired reliability is obtained. A successful architecture must be carefully designed to minimize the overhead of recursive error correction and be able to accommodate some of the most efficient error correcting codes.

3. Efficient quantum resource distribution: The quantum no-cloning theorem [19] (i.e. the inability to copy quantum data) prevents the ability to place quantum information on a wire, duplicate, and transmit it to another location. Each qubit must be physically transported from the source to the destination. This makes each qubit a physical transmitter of quantum information, a restriction which places great constraints on quantum data distribution. Particularly troublesome is moving the qubits over large distances where it must be constantly ensured the data is safe from corruption. One method is to repeatedly error correct along the channel at a cost of additional error correction resources. Another solution is to use a purely quantum concept to implement a long-range wire [10]: teleportation [20], which has been experimentally demonstrated on a very small scale [21, 22, 23]. Teleportation transmits a quantum state between two points without actually sending any quantum data, but rather two bits of classical information for each qubit on both ends.

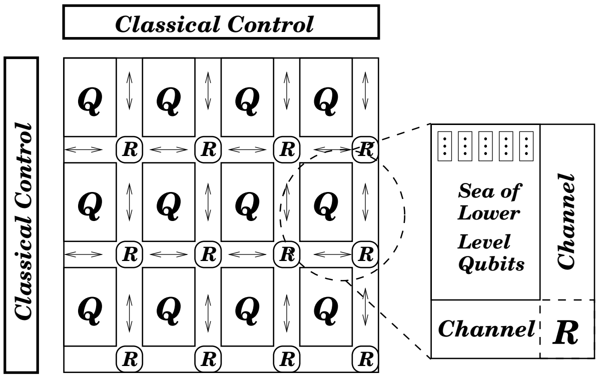

Figure 1. High-Level quantum computer structure, where a full-size computer consists of interconnected logical qubits connected with programmable communication network. The letters *R* denote an integrated switch islands for redirecting quantum data coming from nearby logical qubits or other repeater islands.

This paper introduces and evaluates the design of the Quantum Logic Array (QLA) architecture which takes the following approach to leveraging the three architecture requirements described above:

1 At the lowest level QLA is based on the trapped ion-technology [24, 25, 26], which uses a single trapped atomic ion as a storage for a single unit of quantum data. In particular QLA is based on the highly scalable model of (CCD) style ion-trap quantum information processing architecture proposed by Kielpinski et al [27, 25]. This model consists of ions trapped in interconnected trap arrays and moved from trap to trap to interact [23, 22].

2 We have designed the architecture as a block structure (Figure 1), which fits naturally to quantum error correction, where each building block/tile reflects the error-correction algorithm used. QLA itself is built by tiling these building blocks to form the hierarchies required for larger and more reliable encodings. In addition, QLA invests area in communication channels to allow movement of ions without hindering the parallelism required by fault tolerant structures.

3 By structuring the large-scale model as a datapath oriented block architecture of independent, tightly compact computational units QLA allows us to limit direct ion movement to shorter, local distances within each computational unit. At larger distances (i.e. between computational units) we employ teleportation to avoid mov-

ing data directly over the long channels. Furthermore we couple teleportation with the concept of quantum repeaters [28] to avoid the exponential resource overhead.

The Contributions of this Paper are: 1) We propose the QLA micro-architecture, which is designed for efficient quantum error-correction and error-free long range communication of quantum states. **2)** While teleportation has been proposed as a means of communication, we show the limitations of a simplistic approach using teleportation. We then show how the QLA micro-architecture can be effectively used to overcome these limitations. **3)** To model QLA, we developed ARQ: a scalable quantum architectures simulator that maps quantum applications to fault-tolerant layouts for simulation. ARQ's input is based on the circuit model [29] of quantum computation, which is the most common representation of quantum applications, and allows the tight integration of algorithms and architecture. The complexity of simulating a complete *n*-qubit quantum system grows as $O(2^n)$ on a classical machine. ARQ avoids exponential simulation costs by simulating only a subset of the possible quantum gates, which can be simulated in polynomial time using a mathematical stabilizer formalism - the same formalism at the core of the most efficient quantum error correction codes [30, 31]. **4)** To demonstrate the utility of the QLA, we analytically evaluate its performance when factoring a 128-bit number using Shor's algorithm. We have developed a scheduler to manage the communication issues, using which we determine the bandwidth required to minimize communication overhead. Finally, we show that the QLA, if it were to be implemented using best foreseen ion trap technologies, might allow the implementation of Shor's algorithm to factor a 128-bit number in a time on the order of tens of hours, which is significantly faster than current classical computers might achieve.

Our work complements recent work by Balensiefer in [1], which describes a software tool-chain for ion-trap architectures. However, our focus is on developing a more optimized microarchitecture based upon a more analytic approach, verified through low-level physical simulation. As we shall see, quantum error-correction is a recursive process and low-level simulation is important to account for small factors that accumulate exponentially. Our QLA microarchitecture enables substantial performance improvements critical to supporting full-scale applications such as Shor's factorization algorithm [32]. Balensiefer's work provides the software infrastructure to simplify development of such applications.

The rest of this paper is organized as follows: Section 2 gives a brief overview of trapped ion technology. Section 3 then introduces the QLA micro-processor, followed by its detailed structure and characteristics of its components in Section 4. Section 5 is our evaluation of a large system executing Shor's algorithm. Finally, we offer discussion of future work and conclusions in Sections 6 and 7.

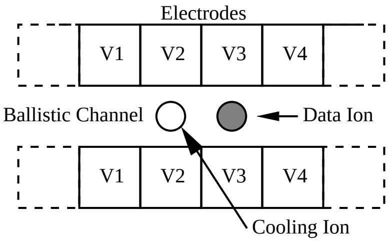

Figure 2. A simple schematic of the basic elements necessary for trapped ion quantum computing.

2. Technology Description

Quantum computers are no longer a fantasy for the future. In particular, quantum ion-trapping technology may potentially lead to a quantum computer with memory size of 50-100 qubits within the next 5 years [33]. While this may seem to be a very small computer, efforts are underway to construct prototypes that will demonstrate the microarchitectural building blocks for a large-scale machine.

A high-level schematic of our ion-trap quantum information processor is shown in Figure 1. The figure shows a number of logical computational units (denoted by the letter *Q*) separated by long range teleportation based communication channels. Each computational unit is a sea of physical atomic ions as shown in Figure 2. The quantum ion-trap processor is surrounded by classical processors, which are used to control the execution of almost everything, from processing quantum measurement information to scheduling of the laser pulses that operate on the ions.

We now take a step back and give a brief overview of the ion-trap technology followed by the expected technology parameters in Subsection 2.2.

2.1. Ion-Traps: a Brief Overview

Ion-trap quantum computation, initially proposed by Cirac and Zoller [24], uses a number of atomic ions that interact with lasers to quantum compute. Qubits are stored in the internal electronic and nuclear states of the ions and the traps themselves are segmented RF Paul traps that allow individual ion addressing (Figure 2). Two ions in neighboring traps can couple to each other forming a linear chain. The vibrational modes of this chain allow a number ions to interact for multi-qubit quantum gates, which together with single qubit rotations yield a universal set of quantum logic. All quantum logic, including measurement, is implemented by applying lasers on the target ions. Individual ions are measured through state-dependent resonance fluorescent readout, where $|1\rangle$ fluoresces weakly and $|0\rangle$ very strongly [34].

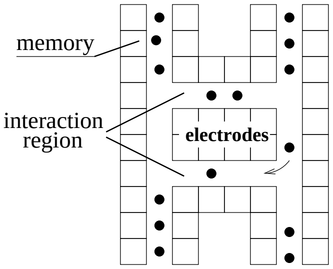

Figure 3. The *QCCD* model proposed by Kielpinski, Monroe, and Wineland in [25]. Ions are *ballistically* shuttled from region to region by changing the trapping voltage potentials.

When ions are manipulated they acquire vibrational heating, which has a negative effect on the gate fidelities. To avoid direct application of cooling lasers on the data ions, which would destroy the quantum information, sympathetic recooling ions are used (as shown in Figure 2), which are always kept at a cooled ground state and are used to absorb much of the vibrational energy from the data ions.

This original proposal, however, does not scale well [35]. As the linear chain size increases the vibrational modes of the ions become harder and harder to identify, thus reducing gate fidelities. Kielpinski, Wineland, and Monroe at NIST have proposed a scalable microarchitecture using interconnected linear trap arrays [25]. Using multiple traps allows for greater control over the logic gates by reducing the size of the linear ion chains. The physical location of each ion within the network is defined by the externally adjustable trapping voltages of the electrodes around the ion. By changing the neighboring voltage potentials ions can be *ballistically* moved from trap to trap, thus allowing the system to handle a very large number of qubits (or ions). This type of ion-trap network is called a Quantum Charge Coupled Device, or simply a *QCCD* (see Figure 3), and has been realized with current alumina micromachining techniques.

Our abstraction of the *QCCD* model assumes that the QLA microarchitecture is a 2-D grid of identical cells, where all cells are attached on the alumina substrate. Cells can contain an ion, electrode, or just be empty to allow a *ballistic channel* for shuttling ions around as shown in Figures 3 or 4. We do not make distinction between memory and interaction regions as in the original proposal, but allow quantum logic, along with qubit initialization to be performed anywhere in the layout. This allows the reuse of ions as the algorithm progresses.

Operation	Time	$P_{current}$	$P_{expected}$
Single Gate	1μs	0.0001	10^{-8}
Double Gate	10μs	0.03	10^{-7}
Measure	100μs	0.01	10^{-8}
Movement	$10ns/\mu$m	$0.005/\mu$m	10^{-6}/cell
Split	10μs		
Cooling	1μs		
Memory time	$10-100$ sec		

Table 1. Column 1 gives estimates for execution times for basic physical operations used in the QLA model. Column 2 gives currently achieved component failure rates $P_{current}$, based on experimental measurements at NIST with $^{9}Be^{+}$ ions, and using $^{24}Mg^{+}$ ions for sympathetic cooling [27, 37]. Column 3 gives projected component failure rates $P_{xpected}$, extrapolated following the ARDA quantum computation roadmap [33], and discussions with the NIST researchers; these estimates are used in modeling the performance of the QLA design.

Ballistic Channels Latency and Bandwidth: Previous work [10] has studied in detail quantum channels which consist of swapping the information from qubit to qubit. The ion-trap case is equivalent if we think of the information being moved on an ion cell by cell along a channel of empty cells. The latency is proportional to the number of cells traversed. If D is the number of cells and T is the time to go from cell to cell, then the total time of the trip is $(\tau + (T \times D))$, where the split time $\tau = 10\mu$s, is the initial cost of starting a movement operation across a channel by splitting the ion from its current chain. Considering a trap of 20μm as suggested in Reference [36] a single trap can be traversed with a time cost of $T = 0.01\mu$s. The independence of the electrode cells from one another allows the ions to move in parallel; thus, pipelining a single channel. In this manner, the ballistic channels provide a bandwidth of $\approx$ 100M qbps (qubits per second).

2.2. Technology Parameters

Table 1 shows a summary of the physical parameters used in our QLA architecture, to model the performance of ion-trap computation. The current experimentally achieved component failure rates are denoted as $P_{current}$, while the expected failure rates, $P_{expected}$, are based on *best-possible* experimental implementations for the technology motivated by recent ion-trap literature [33, 36, 38]. The parameters are justified by the fact that the current challenges with the ion-trap technology are technical; current issues include electrode surface integrity, the structure of the substrate, and precise control of the laser phase, polarization, spatial delivery, and timing stability. Movement errors could be substantially reduced by improving the trap electrode surface integrity [35]. The quality of the trap surface also directly affects movement and gate speed, since its improvement should substantially reduce motional heating. Using semiconductor materials for the trap implementation has been proposed in [25]; this is a technique which should significantly improve the electrode surfaces. Furthermore, precise control of the laser parameters as described in Reference [38] can significantly improve the reliability of the quantum logic gates.

Anticipating advances in ion trap technology and techniques, we choose space and timing parameters for the QLA design as follows. We let the trap separation be $\approx 20\mu$m. Turning a corner at *QCCD* channel intersections is a complicated operation that adds additional motional heating on the ion-qubit. We will let corner-turning speed be equivalent to the time for splitting two ions from a linear chain of 10μs. In addition QLA is designed in such a way that no single gate will require more than two turns when we are using direct ballistic communication, and no turns at all when we are using teleportation.

3. The QLA Architecture

This section provides a brief overview of the QLA architecture (Figure 1). The intent is to introduce the reader to the high level structure of the system. The component details and our low level design decisions are left for Section 4, which follows next.

Block Structure for Error Correction: The underlying structure of QLA is intended for error correction, by far the most dominant and basic operation in a quantum machine [9]. Error correction is expensive because arbitrary reliability is achieved by recursively encoding our qubits at the cost of both exponential resource and operations overhead. Recursive error correction works by encoding N physical ion-qubits into a known highly correlated state that can be used to represent a single logical data bit. This data bit is now at level 1 recursion and will have the property of being in a superposition of “0” and “1” much like a single physical qubit. Encoding once more we can create a *logical qubit* at level 2 recursion with N^2 physical ion-qubits. With each level, L, of encoding the probability of failure of the system scales as $p_0^{2^L}$ as we will see in Section 4, where p_0 is the failure rate of the individual physical components.

The QLA structure fits naturally to quantum error correction because the structure of the building blocks reflects the error-correction algorithm used. Each basic building block represents a single level-one logical qubit as shown in Figure 4. For simplicity, Figure 4 is drawn to show the level 1 blocks of a 3-bit error correcting code, but the structure is easily extended to 7-bit and larger codes [39]. As the figure shows, each building block consists of data ions supported by their cooling ions and trapped between the electrode cells. The investment in communication channels for ballistic ion movement around the physical qubits allows us to limit the high costs of turning. Any two qubit gate at any level of encoding requires at most 2 turns per physical ion in each direction. Furthermore, the adaptability of the QLA design to the application being executed allows us to structure the logical qubits such that they fully comply with the fault-tolerant error correction requirements in References [40] and [41], which demand utilizing maximum parallelism and locality. We empirically demonstrate the fault tolerant property of our design in Section 4.

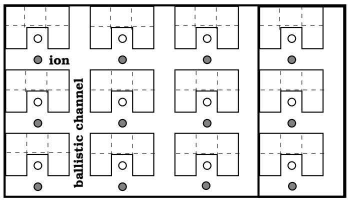

Figure 4. The building blocks of the QLA microarchitecture. Shown are 4 Level 1 building blocks, where the far right side outlines a single block. The circles are data ions (solid) and sympathetic cooling ions clear).

Logical Interconnect: The computational units denoted by the letter *Q* in Figure 1 are in fact encoded logical qubits that represent a single qubit of information whose detailed implementation is described in Section 4.1. Each logical qubit is a regular structure of physical ions as shown in Figure 4 controlled by sequences of *laser pulses*. The logical qubits are positioned on the substrate in a regular array fashion, connected with a tightly integrated repeater-based [28] interconnect as shown in Figure 1. This makes the high-level design very similar to classical tile based architectures. The key difference is that the communication paths must account for data errors in addition to latency. The communication paths are composed of similar physical building blocks as the logical qubits. The integrated repeaters denoted with the letter *R* in Figure 1 are called *teleportation islands* that redirect traffic in the 4 cardinal directions by teleporting data from one repeater to the next. As we will see in Section 5, this interconnect design is one of the key innovative features of our quantum architecture, as it allows us to completely overlap communication and computation, thus eliminating communication latency at the application level of the program.

Programming The Architecture: All scheduling and physical control is performed by the classical processors surrounding the quantum machine. Since physical quantum operations have a latency several orders of magnitude larger than classical operations, a sophisticated classical processor will easily be able to schedule the operations at run-time throughout the execution of the algorithm.

Our general purpose quantum simulator ARQ takes a description of a general quantum circuit with a sequence of quantum gates as an input, maps it onto a specified physical layout, and generates pulse sequence files, which are then executed on the general quantum architecture simulator. For scalability, an actual ion-trap system could manipulate qubits by focusing a small number of lasers through a MEMS mirror array as used in optical routers [42]. The optimization of our algorithms to use the smallest number of lasers, essentially making them more effective for SIMD architectures, is a subject of future research. A tool chain to generate such optimized schedules is also an open area. Our focus is the design of the microarchitecture and its evaluation through hand-optimized applications.

4. Components of the Architecture

This Section describes in more detail the different components of our architecture, along with the design decisions and assumptions we have made in the process of developing QLA. First we describe each logical qubit (Section 4.1), which is followed by a description of the logical interconnect (Section 4.2).

Although the analysis in the following sections becomes somewhat detailed, the key concept is that the structure of QLA supports arbitrary quantum gates such that reliability is increased. We empirically verify the fault tolerant structure of our logical qubit in Subsection 4.1.3; however, at this stage of the design we cannot rely on simulation alone. We use the simulation to validate the analytical intuition that forms the basis of our qubit. We cannot generalize the data to other designs for two reasons: **1)** Data may have multiple inflection points [43] and we might be misled by the analysis of just one point. **2)** We find that level 2 recursion is sufficient, however, it is hard to empirically predict the behavior of a system encoded at higher levels.

4.1. The Logical Qubit Design

The logical qubit design we present is driven by the expected ion-trap parameters (see Table 1, column 4), which place us far below the error threshold required by the threshold theorem, and allow us to optimize for both time and space. Particularly important is the fact that the lifetime of an ion ($\approx$ 10 sec) is far larger than quantum operations which are on the order of tens of microseconds. The relatively low memory error rates allow us to significantly reduce the area of a logical qubit by reducing the parallelism within a single error correction cycle, and the ancillary qubits required by the error correction algorithm.

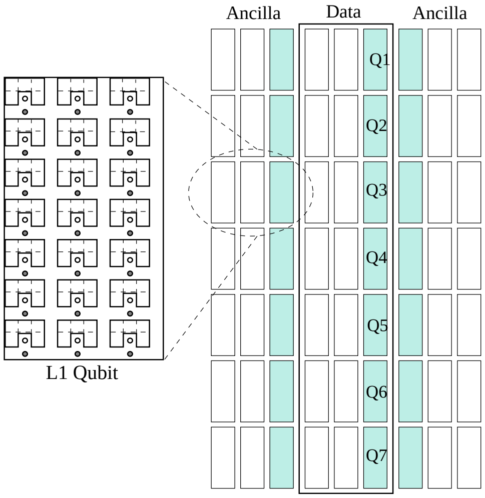

Figure 5. The Logical Qubit: 7 groups of 3 level 1 blocks make a single level 2 logical qubit (middle). The two identical conglomerations on the sides are ancillary blocks used for error correction. The shaded boxes of the level 2 qubit are the encoded data level 1 blocks, which are supported by their respective level 1 ancilla blocks.

Figure 5 shows the full implementation of a level 2 qubit. One of the most important design decisions we have made for each logical qubit, is that it must be a self-contained unit that requires no external *quantum* resources to perform logical gates and state stabilization (i.e. error correction). This will allow an application level compiler to divide the quantum program into distinct data independent threads that are executed on separate computational units, which are simply the logical qubits.

Another important design choice is the error correction code, because it will directly dictate the amount of time each operation requires and the size of the qubit. We choose to model the Steane $[[7,1,3]]$ code, where 7 physical qubits are encoded to form 1 logical qubit that can correct at most $(3-1)/2 = 1$ error. Our choice of the $[[7,1,3]]$ code means that a single data logical qubit at level 2 is built by stacking 7 level 1 blocks. However, each level 1 block must be error corrected at level 1, so to each one we attach two more blocks used as ancilla. To add level 2 error correction we add two more identical ancilla structures at level 2 on both sides of the data logical block. The result is Figure 5. We choose the $[[7,1,3]]$ code because it allows the implementation of a universal set of logical gates *transversally*. This means that a logical quantum bit-flip gate on our qubit can be implemented by applying 49 physical bit-flip gates on the ions, in parallel.

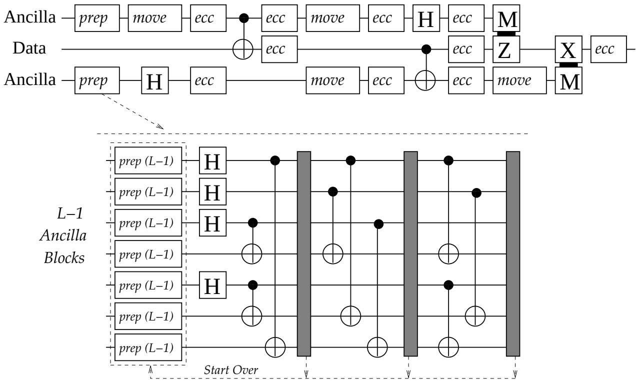

Figure 6. Steane $[\![7,1,3]\!]$ error correction circuit at level L encoding. The top portion is the circuit with one level L data block and two identical ancilla blocks. The boxes represent logical gates or sequences of gates. The `prep` boxes are ancilla preparation, `move` is movement from one block to the next, and `ecc` is error correction of that logical qubit. The bottom portion of the circuit is a zoomed in ancilla preparation stage from [44]. The long shaded boxes are the syndrome extraction for each sub-logical qubit. Movement is implicit in the *CNOT* gates.

4.1.1. Error Correction Latency of our Qubit

Here we estimate the time required for each error correction step at level 2 recursion assuming the expected ion-trap parameters from Table 1. We find that the time to complete a single error correction step at levels 1 and 2 is approximately 0.003 and 0.043 seconds respectively. In our design of the logical qubit we have taken advantage of the low memory failure rate of physical ions to minimize the physical ancilla required at the expense of added error correction time.

The latency times were determined by analyzing the circuit shown in Figure 6,which demonstrates the $[\![7,1,3]\!]$ error correction procedure. In this representation time goes from left to right and various one and two-qubit gates act on each line of qubits. Each line in the circuit denotes a single encoded logical qubit at level 2, and at level 1 in the lower preparation stage.

Computing the Latency: The $[\![7,1,3]\!]$ error correction algorithm [45, 46, 44] consists of extracting a syndrome to determine the location of bit-flip (X) and phase-flip (Z) errors and applying a correction operation based on the extracted syndrome. For each type of error, the syndrome extraction process consists of independently encoding a block of ancilla at the same level as the data qubit and interacting the ancilla and the data. Clearly, the number of ancilla blocks we have affects the parallelism we can explore when extracting syndromes for the two types of error. For example, the level 1 qubit shown on the left of 5 uses 7 ions as data and 7 ions as ancilla, the other 7 are used as verification bits of the encoding. Thus we must extract the two syndromes one after the other. At level 2 however we have ancilla conglomerations on both sides of the data block (see Figure 5) and we can prepare the ancilla blocks in parallel and extract the syndromes in parallel as shown in the circuit of Figure 6.

The $[\![7,1,3]\!]$ error correction circuit in Figure 6 starts with syndrome extraction, which begins with the preparation of the ancilla qubits and ends with the two measurement gates. If a syndrome extraction yields a non-trivial syndrome (i.e. error exists) we repeat the process until we reach two successive agreeing error syndromes. The next step is to correct the error with the appropriate gate followed by a lower level error correction cycle. Equation 1 below, is our estimate for the error correction latency at level L recursion. We have made the following assumptions: **(a)** Two syndromes are extracted in *serial* for both X and Z errors. **(b)** We assume that in the case of a non-trivial syndrome the next extracted syndrome will match it, thus we can proceed with the error correction step. We show this empirically further down. Since our logical qubit at level 2 is equipped with parallel syndrome extraction, assumption (a) makes Equation 1 an overestimate of the final latency:

$$T_{L,ecc} = \begin{cases} 2 \times T_{L,synd}, & \text{Trivial syndrome} \\ 2(2T_{L,synd} + T_1 + T_{L-1,ecc}), & \text{Non-trivial} \end{cases} \tag{1}$$

where $T_{L,synd}$ is the time to extract a syndrome at level L, which is a function of the time to prepare the logical ancilla block. T_1 denotes the time of a logical one-qubit gate, and $T_{L-1,ecc}$ is the time for a lower level error correction step that follows each level L logical gate.

Numerical simulations of a level 2 qubit showed that a non-trivial syndrome was measured for level one with a rate of $3.35 \times 10^{-4} \pm 0.41 \times 10^{-4}$, and for level two at a rate of $7.92 \times 10^{-4} \pm 0.81 \times 10^{-4}$. Our simulations did not yield a syndrome repetition of more than two times before the error correction step. Thus, it is a reasonable assumption that in the case of a non-trivial syndrome we require at most one more syndrome extraction before we are ready to apply the correcting gate. Taking a weighted average of the two cases in Equation 1 we determine a level 2 error correction time of approximately 0.043 seconds, where almost 0.008 seconds is spent in preparation of the logical ancilla.

4.1.2. Qubit Size and Recursion Level

In this subsection we explain why level 2 recursion is sufficient. The level of recursion for each logical qubit is the most crucial assumption for both the performance and the size of our system, since the amount of both computational and physical resources rises exponentially as a function of the recursion level.

A quantum computer running an application of $S = KQ$ elementary steps (or gates), where K is the number of timesteps and Q is the number of logical qubits, will require the elementary component failure rate to be reduced to less than $P_f = 1/S$. To evaluate the expected component failure rate at some level or recursion L, we use Gottesman's estimate for local architectures [40] shown in Equation 2 below.

$$P_f = \frac{1}{cr^2 r^L}(cr^2 p_0)^{2^L} = \frac{p_{th}}{r^L}(p_{th}^{-1} p_0)^{2^L}, \tag{2}$$

where the value for r is the communication distance between level 1 blocks which are aligned in QLA to allow $r =$ 12 cells on average. The threshold failure rate, p_{th}, for the Steane $[\![7,1,3]\!]$ circuit accounting for movement and gates was computed in [41] to be approximately 7.5×10^{-5}. Taking as p_0 the average of the expected failure probabilities given in Table 1, and plugging these numbers into Equation 2, we get an estimated level 2 failure rate of 1.0×10^{-16}. This gives a computer of size $S = KQ = 9.9 \times 10^{15}$ elementary steps. As a simple example, we can consider Shor's factoring algorithm for a 1024-bit number. Employing a circuit description optimized for latency in Reference [47], we find the computer must be of approximate size $S = 4.4 \times 10^{12}$ elementary steps, which is a few orders of magnitude below the computation size attainable with level 2 recursion.

4.1.3. Numerical Analysis of the Logical Qubit

In this subsection we use ARQ to empirically compute p_{th} at level 2 for the QLA logical qubit. Our results, displayed in Figure 7, show that the failure probability of a single one-qubit logical gate rapidly drops to zero at component failure rates lower than $p_{th} = (2.1 \pm 1.8) \times 10^{-3}$. Above this value the rapid decrease in the reliability of our system as recursion increases can be attributed to the additional resource overhead of recursion.

Our estimated threshold failure probability is much higher than the theoretical estimate of 7.5×10^{-5} computed in [41] for several reasons; **1)** The structure of our qubit is optimized for the error correction circuit and may vary for different codes; **2)** The high reliability of ion-trap memory has allowed us to significantly reduce the overall area and ancillary resources required; **3)** The fixed, low movement error probability, and the fact that we made the design decision to never physically move the data, pushed our qubit's threshold closer to the one estimated by Reichardt, 9×10^{-3}, in [44]. We observed no failure at level 2 recursion as the physical component errors approached the expected ion-trap parameters from Table 1, which was expected. Reevaluating Equation 2 with the empirical value for p_{th} we get an estimated level 2 reliability approaching 10^{-21}.

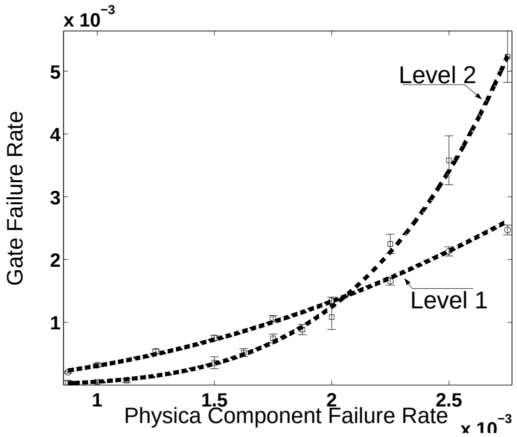

Figure 7. Estimate of the failure probability ($\hat{y}$ axis) of a single logical one-qubit gate followed by recursive error correction procedure at levels 1 and 2. The $\hat{x}$ axis denotes individual physical component failure rates.

Experimental Procedure: To verify that below a certain threshold failure rate, p_{th}, recursion indeed improves the reliability of our logical qubit, we mapped the circuit in Figure 6 exactly to the layout shown in Figure 5 and simulated the execution of a single logical one-qubit gate followed by error correction at recursion levels 1 and 2 respectively. As baseline technology parameters we fixed the movement failure rate to be the expected rate shown in Table 1, but varied the rest of the failure probabilities until we saw a crossing point between the two levels of recursion.The horizontal axis of Figure 7 marks the physical component failure probability and the vertical axis marks the failure probability of the logical gate.

4.2. Logical Qubit Interconnect

At level 2 recursion as described above, our qubit will have dimensions of: (36×147) $cells = 2.11mm^2$ at 20 μm large on each cell side. At this rate we can fit 100 logical qubits per 90nm technology Pentium IV processor, where each such P4 can fit 55 million classical transistors. As we will see in Section 5, to factor a 1024-bit number we may need to communicate over a distance as large as 60 centimeters. Given that such a large chip can be physically realized, there are two ways to transport quantum data at large distances while keeping it protected: **1)** Through channels equipped for repeated error correction of the data; and **2)** To use the concept of *quantum teleportation* [20], which requires the exchange of classical data to recreate the state of the quantum data in its destination. By coupling the teleportation concept with the concept of quantum repeaters [28], we find that we can avoid the high costs of repeated error correction and provide a highly reliable, low-latency, fault-tolerant network interconnection between the logical qubits. In Section 5 we see that for a high level application our network allows the complete overlap between communication and computation. We proceed in this section with a detailed analysis of this network.

Quantum Teleportation: Teleportation begins by preparing two maximally entangled qubits A and B in an Einstein-Podolsky-Rosen (EPR) state [48]: $|\Psi\rangle = |0_A 0_B\rangle + |1_A 1_B\rangle$. Qubit A is sent to the location of the source qubit C and qubit B to C's intended destination. Entangling A and C and measuring them allows us to recreate the state of C over the destination qubit B, where we have only communicated the value of the measurement as classical data. We have effectively teleported C's state over a very large distance without having to move it directly. As a side note, the original states of C and A have been destroyed by the measurement, thus never violating the no-cloning theorem.

The drawback of the teleportation scheme is that we are still physically moving the entangled qubits A and B; however, EPR pairs are replaceable and with enough resources we can establish entanglement between the source and the destination just in time for the communication to be completed. The damaged EPR pairs can be fixed by a process called *entanglement purification* [49, 50], which uses ancillary EPR pairs to distill the good ones from the bad ones. The caveat is that the amount of resources increases exponentially with the EPR separation distance, along with the fact that if the EPR pair becomes too corrupted it may not even lend itself to purification.

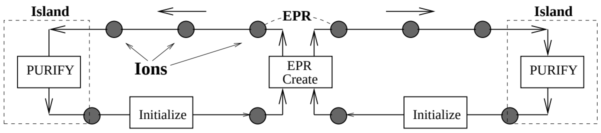

Figure 8. Detail of a channel between two repeater stations. The channel is two-way ballistic transport region, where the EPR pairs are created in the middle and distributed in a pipeline fashion to the two Island/Reapeater stations.

Quantum Repeaters: The EPR transport problem can be solved by combining the concepts of *quantum repeaters* [28] with entanglement purification. The quantum repeaters are islands that are strategically placed in the channels between the logical qubits to limit the distance traveled by each EPR pair (see Figure 1). EPR pairs only travel to two near-by islands, where they can be efficiently purified using the purification protocols with some additional ancillary EPR pairs. To expand a single entangled EPR pair between the source and the destination over the entire channel we use a logarithmic algorithm similar to computing transitive closure. The stages of transport are as follows: **(a)** EPR pairs are created to connect each neighboring repeater station; **(b)** We teleport in parallel across the stations to reduce the amount of connecting EPR pairs by half at each step, but still keep the connection between the source and the destination; **(c)** Successive teleportation steps reduce the EPR pairs by half each time, until we have a single EPR pair connecting the source and the destination in logarithmic number of teleportation hops. Finally we teleport the source qubit to its desired location when a single EPR pair spans the connection channel.

To optimize space and performance we modeled the channels between each island as a two-way ballistic transport region (see Figure 8). Each EPR pair is created in the middle and separated to the two opposing ends. One pair is designated as the data EPR and is continually purified in round-robin pipeline fashion. We assume to have enough ions to handle the maximum amount of required purification steps without having to wait for the creation of new EPR pairs.

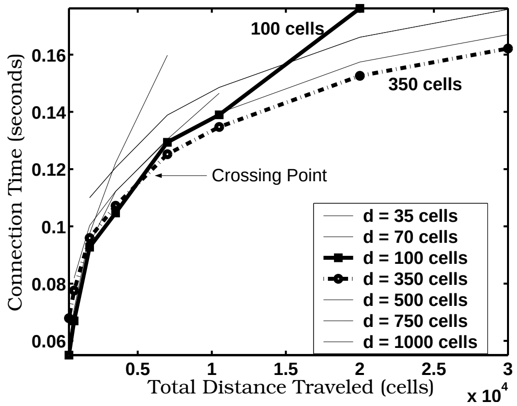

Figure 9. A plot of the total connection time for different island separation distance $d = \{35, 70, 100, 350, 500, 1000\}$ cells between two distant qubits A and B. With each line showing each distance d, we see that island separation of 100 cells is more efficient at distances smaller than 6000 cells (e.g. below ≈ 140 qubits in the $\hat{x}$ direction.) At larger distances separation of 350 cells is preferable.

The teleportation islands are equipped with the capability of being used or not being used. This allows a communication scheduler to pick the optimal inter-island separation for the total distance traveled. Borrowing and adapting the recursive fidelity equations (9,19) given in [28] for the Bennett purification protocol [49], and limiting purification to be only between two adjacent islands we determine optimal separation between two islands to be about 100 cells at distances less than ≈ 6000 cells and about 350 cells at greater distances (see Figure 9). In the $\hat{x}$ direction this amounts to an island at every third and tenth logical qubit respectively. In the $\hat{y}$ direction, however, we place an island at every logical qubit due to the fact that a logical qubit is 147 cells in this direction. The total connection time in Figure 9 was determined by adding enough purification steps between neighboring repeater stations to avoid purification of the final EPR pair between the source and the destination.

5. QLA Performance

In this section we estimate the performance of QLA when executing a general quantum application through the specific example of Shor's factoring algorithm, which is designed to break the widely used RSA public-key cryptosystem. RSA's security lies at the assumption that factoring large integers is very hard, and as the RSA system and cryptography in general have attracted much attention, so has the factoring problem. The efforts of many researchers have made factoring easier for numbers of any size, irrespective of the speed of the hardware. However, factoring is still a very difficult problem. The best classical algorithm known today [51] has complexity of:

$$\exp\left((1.923 + o(1))(\log N)^{1/3}(\log\log N)^{2/3}\right),$$

for an N-bit integer. Using this algorithm Reference [52] has demonstrated the factorization of a 512-bit number in seven calendar months on 300 fast workstations, two SGI Origin 2000 computers, and one Cray C916 Supercomputer - a process which amounts to 8400 MIPS years.

Shor's quantum factoring algorithm [32] allows factoring of large integers in polynomial time. The algorithm works by using a reduction of the factoring problem to finding the period r of the periodic function $f(x) = a^x \bmod M$, where a is a randomly chosen number co-prime to M, x is an integer in $\mathcal{Z}_{2M^2}$, and M is the number being factored. By far the dominant part of the algorithm is this first modular exponentiation portion, which computes $f(x)$ in superposition, over all values of x. A second part is the quantum Fourier transform (QFT), which finds the period of $f(x)$ from the results previously computed. The overview of the cost for factoring different N-bit numbers is given in Table 2. Area numbers assume scaling along the ARDA roadmap [33] to 20 μm traps, which is also the assumed size of each cell in the QLA layout.

	N=128	N=512	N=1024	N=2048
Logical Qubits	37,971	150,771	301,251	602,259
Toffoli Gates	63,729	397,910	964,919	2,301,767
Total Gates	115,033	1,016,295	3,270,582	11,148,214
Area(m^2)	0.11	0.45	0.90	1.80
Time(days)	0.9	5.5	13.4	32.1

Table 2. System numbers for Shor's algorithm for factoring an N-bit number using the circuit descriptions of [53, 47] and the QLA microarchitecture model. The QLA chip area is determined by the number of logical qubits and channels (qubits: 147×36 cells with added 11 and 12 cells for the channels, where each cell is 20μm large on each side.

Since quantum modular exponentiation is the most computationally intensive component of Shor's algorithm, many papers address the need to design efficient quantum arithmetic circuits. The design of quantum adders is specially interesting since modular exponentiation consists of modular multiplication, which itself can be divided into additions. We consider a quantum logarithmic-depth quantum carry lookahead adder (QCLA) [53] as a component to perform quantum modular exponentiation. The QCLA is based on ideas derived from the classical lookahead adder. It can perform an n qubit addition with a latency of $4\log_2 n$ Toffoli gates, 4 CNOT's and 2 NOT's, and is an adder chosen from [53] to be most optimized for time of computation rather than system size.

We leverage previous research [47] that explores various algorithms and techniques to reduce the latency of a complete quantum modular exponentiation. The latency of modular exponentiation is computed by the equation $MExp = IM \times MAC \times (QCLA + ArgSet) + 3p \times QCLA$, where IM is the number of calls to the multiplier, MAC is the number of calls to the adder block required to perform an n-bit modulo multiplication. ArgSet refers to the technique of indirection which allows us to reduce the number of multiplications. Finally, p is the number of extra qubits required by the adders for optimization, and $QCLA$ is the depth the QCLA circuit.

We take this a step further by considering the effects of fault-tolerance and qubit movement. As Table 2 shows the dominant gate in the modular exponentiation procedure is the Toffoli gate, which is a three qubit controlled-controlled-NOT gate. A fault-tolerant construction of this gate using a universal one and two-qubit gate basis requires 6 additional logical ancilla qubits. The fault-tolerant Toffoli circuit we analyze, which can be constructed following

[54, 55], takes into consideration both the fault-tolerant Toffoli gate and the ancilla preparation required. The cost of a fault-tolerant Toffoli is much greater than that of one two-qubit or a single-qubit gate. The preparation of the ancilla qubits is an involved process of 15 timesteps repeated three times. However each Toffoli gate is performed on an independent set of logical qubits; thus the ancilla preparation of each successive Toffoli can be overlapped in most cases with the execution of the previous Toffoli gates.

When we consider concurrency in the algorithm as a whole, it can be seen that we can easily perform the required two qubit gates in parallel with the Toffoli ancilla preparation. Thus, we only consider the cost of performing fault tolerant Toffoli gates in our overall time evaluation for the modular exponentiation. The ancilla preparations of each Toffoli gate can be overlapped; however, in many Toffoli's one of the three qubits involved shares its ancilla with a previous Toffoli. Therefore each Toffoli will contribute approximately 15 error correction steps for the ancilla preparation and 6 error correction cycles to finish the gate. A single time-step is defined by an error correction cycle since the qubits involved at each logical gate must be error corrected each time.

The critical component for the success of the whole design is the cost of communication between logical qubits. We have made a design decision that ballistic transport must be used for moving ions within a logical qubit, and teleportation will be preferred when moving across larger distances in order to keep the failure rate due to movement below the threshold amount. The teleportation protocol analyzed maintains constant cost in the face of increasing distance and hence is a critical weapon in our armory. Since EPR pairs are required for teleportation, we can reduce communication costs to a minimum if we have the required number of EPR pairs available at a logical qubit at the same time that it is ready to move. Fortunately, this is possible because of the high cost of error correcting the logical qubits. We can create, purify and transport the required EPR pairs to their respective qubits while they are undergoing error correction. But can this be done at a large scale?

To answer this question, we developed a tool to schedule the movement of EPR pairs in QLA. We assigned one channel to carry the created EPR pairs to their destinations and another channel to return the used EPR pairs. Within each channel, the EPR pairs are pipelined. We define the bandwidth of QLA's communication channels as the number of physical channels in each direction. The distance between each Teleportation Island was fixed at 100 cells. The goal of our scheduler then, is to find paths between logical qubits to transport all the required EPR pairs within the time it takes to perform a level 2 error correction.

The scheduler is heuristic greedy scheduler that scalably achieves an average of $\sim 23\%$ aggregate bandwidth utilization on our implementation of the Toffoli gate. It works by grabbing all available bandwidth whenever it can. However, if this means that the scheduler cannot find the necessary paths, it will back off and retry with a different set of start and end points. A simple approach to doing a two qubit gate between logical qubits A and B would be as follows: teleport A to B's physical location, perform the gate and teleport it back. An optimization that the scheduler incorporates is that it only moves logical qubit A back if necessary. As a result, the logical qubits *drift* from one location to another. This adds a level of complexity to the scheduler, but at the same time reduces the amount of movement that the qubits are subjected to.

With all the above considerations in the scheduler, we found that given two channels in each directions (bandwidth of 2), we could schedule communication such that it always overlapped with error correction of the logical qubits. The end result being reliable movement over arbitrary distances with minimal overhead.

The total time for modular exponentiation will be dominated by error correction of the logical qubits within a fault-tolerant Toffoli gate. For a 128 bit number, modular exponentiation requires 63730 Toffoli gates with 21 error correction steps per Toffoli. The error correction steps of the entire algorithm amount to ($21 \times 63730 + \text{QFT} = 1.34 \times 10^6$). Since 0.043 seconds are required to perform one error correction at level 2 recursion, it will take approximately 16 hours to complete the factorization of a 128 bit number. However, assuming success of all the gates, the circuit is repeated on average 1.3 times [56], so the total time to factor a 128 bit number would be around 21 hours. Similar calculations lead to the execution times of the factorization of larger integers shown in Table 2; however, the sheer sizes of the ion-trap chips required make the physical realization of such a systems a considerable engineering challenge, which may be impractical for $N > 128$, with current single chip technology.

6. Future Work

The QLA architecture leverages current quantum architectures research; however, its development must also leverage the vast amount of knowledge and research from classical architectures. Several critical issues quickly come to mind for the advancement of the quantum architecture: relaxing the technology restrictions; management of classical resources; and finally reducing the area of the architecture.

Relaxing the Technology Restrictions: Relaxing the technology restrictions will lead to quicker realization of the QLA microarchitecture. The assumption of the expected ion-trap failure probabilities is not unrealistic as base parameters for a future system such as this one. Careful simulations, however, of various noise models and encodings must be conducted to determine how far we can relax this assumption in order to have a system still capable of relevant quantum computing with exponential speedup over classical machines.

Management of Classical Resources: The success of the physical implementation of the QLA model rests upon the realization of the classical mechanisms that control the execution of a quantum algorithm. Some of these mechanisms include; the control of lasers for precise manipulation of thousands of logical qubits; the amount of laser power possible; the number of photodetectores required for measurement; and even the wiring of the electrodes used for trapping the physical ions. Classical resources must be optimized both in quantity and usage complexity through clever scheduling and representation of quantum operations. Furthermore, the inherent parallelism in quantum computation along with the fundamentally different operations structure has the potential to create a wide variety of interesting and very difficult compiler design problems.

Computer Area: According to the results in Table 2, the area of the ion-trap chip for even the factoring of a 128-bit number is roughly 0.45 square meters. This amounts to a chip size of 33 centimeters at each edge if we assume a square chip. This is a substantial fabrication and yield challenge. QLA offers an inherent redundancy within itself, which we can explore to raise the yield. This arises from the fact that all logical qubits and channels are identical in both their structure and ability to support different functionalities. Defects can be diagnosed and masked out in software running on our classical control processor. The fabrication challenges, however, suggest that a multi-chip solution for solving such large problems is desirable. Although, experimental progress has been made in this direction, [57, 58, 59], the detailed analysis of the design and performance of such systems remains as an area for future research.

7. Concluding Remarks

This paper has introduced the QLA, which is a quantum computer architecture for trapped ions, designed for efficient error-correction and error-free communication, over arbitrary on-chip distances. Our designs are validated by analytical reasoning and also by simulation. We have emphasized the importance of a datapath oriented large-scale quantum architecture for solving realistic problems, and shown how the QLA achieves such design goals. In addition, we have introduced ARQ, a tool used to map quantum applications to fault-tolerant architectures.

Finally, we have shown how the QLA architecture design methodology potentially scales, conceivably allowing a system of 7×10^6 physical ions to be able to implement Shor's algorithm to factor a 128-bit number within 1 day; such performance assumes aggressive technology parameters which are not currently achieved, but are believed to be within reach of present experimental techniques. The QLA architecture should thus provide vital insight and motivation, from a systems level perspective, to physicists involved in actually building a large-scale quantum computer. For architects, the QLA forms a technically sound base, that can be used to confidently study interesting issues in quantum architectures and to work towards more efficient, reliable, and scalable quantum computers.

Acknowledgements: This work is supported in part by the DARPA QUIST program, in part by a NSF CAREER grant and a UC Davis Chancellor's Fellowship awarded to Fred Chong, and in part by the NSF Center for Bits and Atoms at MIT

References

[1] S. Balensiefer, L. Kregor-Stickles, and M. Oskin, "An evaluation framework and instruction set architecture for ion-trap based quantum micro-architectures.," *ISCA-32; Madison, WI* , 2005.

[2] D. Deutsch, "Quantum computational networks," *Proc. R. Soc. Lond.* **A 400**, pp. 97–117, 1985.

[3] L. Grover *Symposium on Theory of Computing (STOC 1996)* , pp. 212–219.

[4] A. M. Childs, E. Farhi, and J. Preskill, "Robustness of adiabatic quantum computation," *Phys. Rev. A* **65**, 2002.

[5] I. L. Chuang, "Quantum algorithm for clock synchronization," *Phys. Rev. Lett.* **85**, p. 2006, 2000.

[6] C. Bennett and G. Brassard, "Quantum cryptography: Public key distribution and coin tossing," *IEEE International Conference on Computers, Systems, and Signal Processing* , pp. 175–179, 1984.

[7] W. van Dam and G. Seroussi, "Efficient quantum algorithms for estimating gauss sums," *E-Print: quant-ph/0207131* , 2002.

[8] S. Hallgren, "Polynomial time quantum algorithms or pell's equation and the principal ideal problem," *Symposium on Theory of Computing (STOC 2002)* , pp. 653–658.

[9] M. Oskin, F. Chong, and I. Chuang, "A practical architecture for reliable quantum computers," *IEEE Computer* , January 2002.

[10] M. Oskin, F.T.Chong, and I. Chuang, "Building quantum wires: The long and the short of it," *ISCA-30, San Diego, CA* , 2003.

[11] D. Copsey, M. Oskin, T. Metodi, F. Chong, and I. Chuang, "The effect of communication costs in solid-state quantum architectures," *SPAA 2003; San Diego, CA* .

[12] D. P. DiVincenzo, "The physical implementation of quantum computation," *Fortschr. Phys.* **48**, pp. 771–783, 2000.

[13] P. W. Shor, “Scheme for reducing decoherence in quantum computer memory,” *Phys. Rev. A* **54**, p. 2493, 1995.

[14] A. Steane, “Error correcting codes in quantum theory,” *Phys. Rev. Lett* **77**, pp. 793–797, 1996.

[15] E. Knill and R. Laflamme, “A theory of quantum error-correcting codes,” *Phys. Rev. A* **55**, pp. 900–911, 1997.

[16] D. Gottesman, “A class of quantum error-correcting codes saturating the quantum hamming bound,” *Phys. Rev. A* **54**, p. 1862, 1996.

[17] D. Aharonov and M. Ben-Or, “Fault tolerant computation with constant error,” *Symposium on Theory of Computing (STOC 1997)* , pp. 176–188.

[18] A. Y. Kitaev, “Quantum error correction with imperfect gates,” *3rd Int. Conf. of Quantum Communication and Measurement* , pp. 181–188, 1997.

[19] W. Wootters and W. Zurek, “A single quantum cannot be cloned,” *Nature* **299**, pp. 802–803, 1982.

[20] C. H. Bennett and et. al., “Teleporting an unknown quantum state via dual classical and EPR channels,” *Phys. Rev. Lett.* **70**, pp. 1895–1899, 1993.

[21] D. Bouwmeester and et. al., “Experimental quantum teleportation,” *Nature* **390**, pp. 575–579, 1997.

[22] M. Riebe, H. Haffner, C. Roos, and et. al., “Deterministic quantum teleportation with atoms,” *Nature* **429**(6993), pp. 734–737, 2004.

[23] M. Barrett, J. Chiaverini, T. Schaetz, J. Britton, and et. al., “Deterministic quantum teleportation of atomic qubits,” *Nature* **429**, 2004.

[24] J. I. Cirac and P. Zoller, “Quantum computations with cold trapped ions,” *Phys. Rev. Lett* **74**, pp. 4091–4094, 1995.

[25] D. Kielpinski, C. Monroe, and D. Wineland, “Architecture for a large-scale ion-trap quantum computer,” *Nature* **417**, pp. 709–711, 2002.

[26] J. Porto, S. Rolston, T. Laburthe, C. Williams, and W. Phillips, “Quantum information with neutral atoms as qubits,” *Phil. Trans. R. Soc. Lond.* **A361**, pp. 1417–1427, 2003.

[27] D. Wineland and et al., “Experimental issues in coherent quantum-state manipulation of trapped atomic ions,” *Journal of Research of NIST* **103**, pp. 259–328, 1998.

[28] W. Dur, H. J. Briegel, J. I. Cirac, and P. Zoller, “Quantum repeaters based on entanglement purification,” *Phys. Rev.* **A59**, p. 169, 1999.

[29] V. Vedral, A. Barenco, and A. Ekert, “Quantum networks for elementary arithmetic operations,” *Phys. Rev.* **A54**, p. 147, 1996.

[30] D. Gottesman, “The heisenberg representation of quantum computers,” *Hobart, Group theoretical methods in physics, E-Print: quant-ph/9807006* , pp. 32–43, 1998.

[31] S. Aaronson and D. Gottesman, “Improved simulation of stabilizer circuits - in preparation, 2004,” *E-Print: quant-ph/0406196* , 2004.

[32] P. Shor, “Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer,” *35’th Annual Symposium on Foundations of Computer Science* , pp. 124–134, 1994.

[33] D. Wineland and T. Heinrichs, “Ion trap approaches to quantum information processing and quantum computing,” *A Quantum Information Science and Technology Roadmap* , p. URL: http://quist.lanl.gov, 2004.

[34] E. Hahn, “Spin echoes,” *Phys. Rev.* **80**, pp. 580–594, 1950.

[35] M. A. Rowe and et. al., “Transport of quantum states and separation of ions in a dual rf ion trap,” *Quant. Inf. Comp.* **2**, pp. 257–271, 2002.

[36] A. Steane, “How to build a 300 bit, 1 gop quantum computer,” *E-Print: quant-ph/0412165* , 2004.

[37] D. Leibfried and et al., “Experimental demonstration of a robust, high-fidelity geometric two ion-qubit phase gate,” *Nature* **422**, pp. 412–415, 2003.

[38] R. Ozeri and et. al., “Hyperfine coherence in the presence of spontaneous photon scattering,” *E-Arxiv: quant-ph/0502063* , 2004.

[39] A. M. Steane, “Quantum computer architecture for fast entropy extraction,” *E-Print: quant-ph/0203047* , 2002.

[40] D. Gottesman, “Fault tolerant quantum computation with local gates,” *Journal of Modern Optics* **47**, pp. 333–345, 2000.

[41] K. Svore, B. Terhal, and D. DiVincenzo, “Local fault-tolerant quantum computation,” *E-Print: quant-ph/0410047* , 2004.

[42] D. J. Bishop, C. R. Giles, and G. P. Austin, “Lucent LambdaRouter: MEMS technology of the future here today,” *IEEE Commun. Mag.* **40**, pp. 75–79, March 2002.

[43] K. Svore, A. Cross, I. Chuang, and A. Aho, “Pseudothreshold or threshold? - more realistic threshold estimates for fault-tolerant quantum computing,” *E-Print: quant-ph/0508176* , 2005.

[44] B. W. Reichardt, “Improved ancilla preparation scheme increases fault-tolearant threshold,” *E-Print: quant-ph/0406025* , 2004.

[45] A. M. Steane, “Space, time, parallelism and noise requirements for reliable quantum computing,” *Fortsch. Phys.* **46**, pp. 443–458, 1998.

[46] A. M. Steane, “Overhead and noise threshold of fault-tolerant quantum error correction,” *E-Print: quant-ph/0207119* , 2002.

[47] R. V. Meter and K. M. Itoh, “Fast quantum modular exponentiation,” *E-Print: quant-ph/0408006* , 2004.

[48] J. S. Bell, “On the Einstein-Podolsky-Rosen paradox,” *Physics* **1**, pp. 195–200, 1964.

[49] C. Bennett and et. al., “Purification of noisy entanglement and faithful teleportation via noisy channels,” *Phys. Rev. Lett.* **76**, p. 722, 1996.

[50] D. Deutsch, A. Ekert, R. Jozsa, C. Macchiavello, S. Popescu, and A. Sanpera, “Quantum privacy amplification and the security of quantum cryptography over noisy channels.,” *Phys. Rev. Lett.* **77**, pp. 2818–2821, 1996.

[51] J. Buhler, H. Lenstra, and C. Pomerance, “Factoring integers with the number field sieve,” *Pages 50-94 in The Development of the Number Field Sieve, volume 1554 of Lecture Notes in Mathematics* **Springer-Verlag, Berlin**, 1994.

[52] S. Cavallar and et.al., “Factorization of a 512-bit rsa modulus,” *Proceedings Eurocrypt 2000* **Springer-Verlag**, pp. 1–17, 2000.

[53] T. Draper, S. Kutin, E. Rains, and K. Svore, “A logarithmic-depth quantum carry-lookahead adder,” *E-Print: quant-ph/0406142* , 2004.

[54] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, Cambridge, UK, 2000.

[55] A. Steane, “Efficient fault-tolerant quantum computing,” *Phys. Rev. Lett.* **78**, pp. 2252–2255, 1997.

[56] A. Ekert and R. Jozsa, “Quantum computation and shor’s factoring algorithm,” *Reviews of Modern Physics* , pp. 733–753, July 1996.

[57] C. Cabrillo and et. al., “Creation of entangled states of distant atoms by interference,” *Phys. Rev. A* **59**, pp. 1025–1033, 1999.

[58] L. Duan, M. Lukin, J. Cirac, and P. Zoller, “Long-distance quantum communication with atomic ensembles and linear optics,” *Nature* **414**, p. 413, 2001.

[59] B. Blinov, D. Moehring, L. Duan, and C. Monroe, “Observation of entanglement between a single trapped atom and a single photon,” *Nature* **428**, pp. 153–157, 2004.